(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 632 367 A1**

(12) EUROPEAN PATENT APPLICATION

(21) Application number: **94110018.2**

(51) Int. Cl.[6]: **G06F 3/06**

(22) Date of filing: **28.06.94**

(30) Priority: **30.06.93 US 85483**

(43) Date of publication of application:
04.01.95 Bulletin 95/01

(84) Designated Contracting States:
DE FR GB

(71) Applicant: **MICROSOFT CORPORATION**
One Microsoft Way
Redmond,
Washington 98052-6399 (US)

(72) Inventor: **Zbikowski, Mark**
15817 N.E. 178th Place
Woodinville,
Washington 98072 (US)
Inventor: **Berkowitz, Brian T.**
3912-142nd Place N.W.
Bellevue,
Washington 98007 (US)
Inventor: **Ferguson, Robert I.**
2910-9 th Avenue West
Seattle,
Washington 98119 (US)

(74) Representative: **Patentanwälte Grünecker, Kinkeldey, Stockmair & Partner**
Maximilianstrasse 58
D-80538 München (DE)

(54) Meta-data structure and handling.

(57) A file system stores data and meta-data in a like fashion. The lowest level of stored file data on disk is a stream, which constitutes a logically contiguous group of bytes. Related streams, such as found in a file, a directory or a subdirectory, are stored in a variable-sized onode data structure. Variable-sized onode data structures are stored in an array of fixed-sized buckets of disk space. Related onode data structures are stored within catalog data structures. The catalog data structures are stored within the array of fixed-sized buckets of disk space. The array of buckets of fixed-sized buckets of disk space is, in turn, stored as a stream.

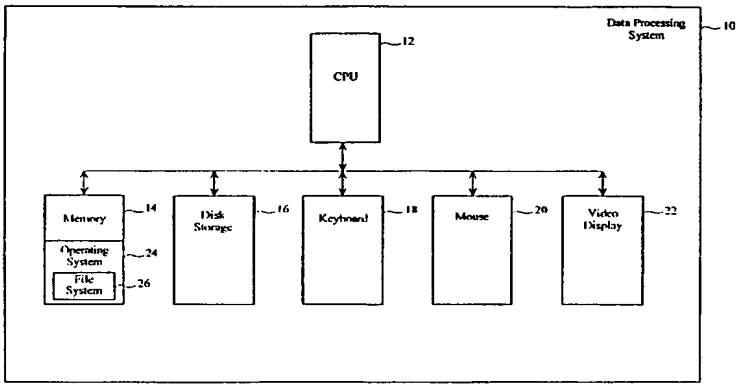

Figure 1

Technical Field

The present invention relates generally to data processing systems and, more particularly, to storage of data on disk by a file system.

Background of the Invention

Conventional file systems have had difficulty in storing file data on disk in an efficient manner. Many conventional systems have adopted an approach wherein all data is stored in a single-sized storage unit on disk. Unfortunately, this approach does not efficiently store file data on disk. In particular, file data may vary in size and, thus, may not be well-matched for the predetermined storage unit size. Other conventional systems have provided a user with an option of adopting one of numerous different formats. The decision as to which format to adopt must be made before the file data is available to the user: As a result, the choice of format by the user is merely speculative and often does not correspond well with the actual file data. As a result, the file data is often inefficiently stored.

In conventional operating systems, each file is allocated a fixed number of blocks of disk space for storing file data and control information about the file. The blocks are fixed-sized so that the units may be quickly allocated and deallocated. The control information for the files and the file data are often variable-sized, which poses at least two problems. First, when the file data and/or control information is not large enough to fill a block, disk space within the allocation unit is wasted. Second, when the file data and/or control information is too large to be stored in a single block, it must be stored in multiple blocks and multiple pointers are maintained to specify where the data is stored. Maintaining and using such pointers is often quite cumbersome.

In conventional file systems, data and meta-data (i.e., data that describes the storage of other data) have been treated as distinct entities. In particular, data and meta-data have been stored in different formats in conventional file systems. Moreover, separate tools have been provided for acting on data and meta-data. This fragmented view of data and meta-data has resulted in added overhead and complexity.

Summary of the Invention

In accordance with the first aspect of the present invention, a method is practiced in a data processing system having disk storage and a processing means for running an operating system. In this method, the data is stored in the disk storage in a first variable-sized stream data structure. Meta-data is stored in the disk storage in a second variable-sized stream data structure. A stream descriptor is stored for each of the stream data structures. The stream descriptor includes a type identifier that identifies how the stream data structure is stored on disk in disk storage.

In accordance with another aspect of the present invention, streams of data are stored in stream data structures in the disk storage. The streams of data are formed from logically contiguous bytes of data. A stream descriptor is stored for each stream data structure in the disk storage. Each stream descriptor includes a type identifier that describes how the stream data structure is stored in the disk storage. The stream data structures which store related data are stored in a first variable-sized data structure along with their associated stream descriptors.

In accordance with yet another aspect of the present invention, streams of data are stored in the disk storage in stream data structures. A stream descriptor is stored for each stream data structure on the disk storage. Each stream descriptor includes a type identified that describes how the stream data structures are stored in the disk storage. Stream data structures which store related data are stored in the disk storage along with their associated stream descriptors in a variable-sized data structure. Groups of the variable-sized data structures which store related data are stored in a catalog data structure.

In accordance with yet another aspect of the present invention, a method is practiced in a data processing system having disk storage. In this method, streams of data are stored in the disk storage in stream data structures. The streams are formed by logically contiguous bytes of data. A stream descriptor is stored for each stream data structure in the disk storage. The stream data structures which store related data are stored along with their associated stream descriptors in respective variable-sized data structures in the disk storage. Each variable-sized data structure has an associated identifier. The variable-sized data structures are stored in an array of fixed-sized identifiable buckets of disk space in the disk storage. A mapping structure holding entries that specify a bucket identifier for a bucket in the array is stared in the disk storage. The entries are indexed by the identifiers of the variable-sized data structures. The mapping structure is used to locate one of the variable-sized data structures in the array, given its identifier.

Brief Description of the Drawings

Figure 1 is a block diagram of a data processing system for practicing a preferred embodiment of the present invention.

Figure 2 is a diagram of the format for a stream descriptor that is used in the preferred embodiment of the present invention.

Figure 3 is a diagram of a stream descriptor for a tiny stream in accordance with the preferred embodiment of the present invention.

Figure 4 is a diagram of a stream descriptor for a small stream in accordance with the preferred embodiment of the present invention.

Figure 5 is a diagram of a stream descriptor for a large stream in accordance with the preferred embodiment of the present invention.

Figure 6 is a diagram of a stream descriptor for a compressed stream in accordance with the preferred embodiment of the present invention.

Figure 7 is a diagram of a stream descriptor for an encrypted stream in accordance with the preferred embodiment of the present invention.

Figure 8 is a diagram of a stream descriptor for a small scale transaction in accordance with the preferred embodiment of the present invention.

Figure 9 is a diagram of a stream descriptor for replicated data in accordance with the preferred embodiment of the present invention.

Figure 10 is a diagram of a field descriptor in accordance with the preferred embodiment of the present invention.

Figure 11 is a diagram of an onode in accordance with the preferred embodiment of the present invention.

Figure 12 is a diagram of streams that hold property information about the onode of Figure 12.

Figure 13 is a diagram of a bucket array in accordance with the preferred embodiment of the present invention.

Figure 14 is a diagram of a work ID mapping array and an onode bucket array in accordance with the preferred embodiment of the present invention.

Figure 15 is a diagram of an object store catalog in accordance with the preferred embodiment of the present invention.

Detailed Description of the Invention

The preferred embodiment of the present invention provides a file system that stores both data and meta-data on disk as groups of "streams." A "stream" is a logically contiguous, randomly addressable, variable-sized array of bytes of data that serves as the logical unit of storage on the disk. Most programmatic access to data in files is made through streams. Each stream may be stored on the disk in one of a number of different representations. Each of the different representations is well suited for a particular size and use of the stream. Accordingly, each stream is stored in a representation on disk that is best suited for its size.

Each stream has a stream descriptor associated with it and which is stored in a file system control structure. The stream descriptor is used to access the stream and to obtain information about the stream. The stream descriptor provides a description of the representation in which the data of the streams is stored. Streams of data that hold related data are encapsulated into data structures known as onodes. An onode is a variable-sized structure that is roughly analogous to a file, a directory or a subdirectory. Groups of related onodes are, in turn, stored in data structures known as object store catalogs or object stores. Thus, data and meta-data (such as the object store catalogs) are described in a like fashion in a hierarchy which will be described in more detail below.

Figure 1 is a block diagram of a data processing system 10 for practicing the preferred embodiment of the present invention. Although the system 10 of Figure 1 is a single processor system, those skilled in the art will appreciate that the present invention may also be practiced in a multiple processor system, such as a distributed system. The data processing system 10 includes a central processing unit (CPU) 12, a memory 14, disk storage 16, a keyboard 18, a mouse 20, and a video display 22. The disk storage 16 may include hard disks and other types of disk storage devices. The keyboard 18, mouse 20, and video display 22 are conventional input/output devices.

The memory 14 holds a copy of an operating system 24, including a file system manager 26 for managing files stored in the system. The operating system 24 may be an object-oriented operating system. The preferred embodiment of the present invention described herein is implemented as part of the operating system 24. Although the preferred embodiment of the present invention is described as being part of the operating system 24, those skilled in the art will appreciate that the present invention may alternatively be implemented in other types of code that are separate from the operating system.

As mentioned above, streams are available in a number of different representations in the preferred embodiment of the present invention. In order to understand the different representations of streams, it is helpful to review the format of the stream descriptors that are provided for the streams. Figure 2 is a diagram of the format of a stream descriptor 28. The stream descriptor 28 is capable of describing each of the different representations of streams that are available in the preferred em-

bodiment of the present invention. The stream descriptor 28 includes three fields: a size field 30, a type field 32 and a description field 34. The size field 30 holds a value that specifies the size of the stream in bytes. The type field 32 specifies the type of the stream, and the description field 34 holds a description of the stream (i.e., the form of the description field 34). The values held in fields 30, 32 and 34 vary with the representation of the associated stream (as will be described in more detail below).

A "tiny stream" is a first representation of a stream that is available in the preferred embodiment of the present invention. The tiny stream is used to store data that is very small in size relative to the allocation unit of the storage medium (i.e., a disk in the disk storage 16). The "allocation unit" of the storage medium refers to the basic unit of disk memory space in disk storage 16 that is allocated for storing files. For instance, in FAT-based file systems, the minimum allocation unit is a sector. Unfortunately, a sector is often much larger than stream data produced in the system 10. Figure 3 depicts the format of the stream descriptor 28 for the tiny stream. The size field 30 holds a value that specifies the size of the stream, and the type field 32 specifies that the stream is a tiny stream. The description field 34 holds the data of the stream and, thus, provides an immediate representation of the data of the stream. This immediate representation provides a very efficient means for storing small amounts of data. In particular, the data is integrated directly into the stream descriptor so that it may be easily and quickly accessed.

Another representation that is available in the preferred embodiment of the present invention is a small stream. The "small stream" is a stream that is stored in a single extent of data. An extent is a variable-sized contiguous run of allocation units. The stream data is stored in an extent because it is too large to store directly in the stream descriptor 28. The format of the stream descriptor 28 for the small stream is shown in Figure 4. The type field 32 specifies that the associated stream is a small stream, and the description field 34 holds an extent descriptor 36 that describes an extent 42 in which the data of the stream is stored. The extent 42 is stored on a disk in the disk storage 16. The extent descriptor 36 includes two subfields 38 and 40. Subfield 38 holds a value that specifies the length of the extent 42, and subfield 40 holds the disk address of the extent (i.e., where the extent is located in the logical address space of the disk).

A third representation that is available in the preferred embodiment of the present invention is a "large stream." The large stream is a stream stored in multiple extents. The large stream is appropriate for storing a stream having a large amount of data. Figure 5 depicts the format of the stream descriptor 28 for such a large stream. Type field 32 specifies that the stream is a large stream. The description field 34 holds a second stream descriptor 43 that describes a stream 44 holding extent descriptors. This second stream descriptor 43 describes a tiny stream and includes a description field 34' that holds the stream 44 of extent descriptors. The extent descriptors 36', 36'' and 36''' have the same format as the extent descriptor 36 that was described with reference to Figure 4. As a result, multiple extents 42', 42'' and 42''' are described by a single stream 34'. If the number of extent descriptors becomes too large, the second stream descriptor 43 may describe a small stream rather than a tiny stream. Moreover, if the number of extent descriptors becomes too large for a small stream, the second stream descriptor 43 may describe a large stream. Large streams are generally used where large contiguous blocks of disk space are not available. The large stream facilitates the storage of large amounts of data as a single stream in extents that may be dispersed about the disk. As a result, large streams scale well as disks become more fragmented and the stream grows.

As the size of a stream grows, the representation of the stream is promoted up a hierarchy of stream representations to facilitate efficient storage of the stream. The hierarchy includes the tiny stream, the small stream and the large stream. A stream may be promoted from a tiny stream to a small stream and then to a large stream. In general, as was mentioned above, the most appropriate representation is chosen for a stream based upon the amount of data and the amount of fragmentation that are included in the stream.

The above description has described the four basic types of streams available in the preferred embodiment of the present invention. The type field 34 of stream descriptor 28 may also be utilized to specify special descriptions of data stored within a stream. Figure 6 is an example of the format of the stream descriptor 28 when a stream holds compressed data. The type field 32 holds a value specifying that the data of the stream is compressed, whereas the description field 34 holds a stream descriptor for the compressed data. The stream descriptor held in the description field 34 may be a tiny stream, a small stream, or a large stream, depending upon the amount of data included in the stream.

Figure 7 depicts the format of a stream descriptor 28 when the stream descriptor describes a stream of encrypted data. Type field 32 holds a value which specifies that the stream holds encrypted data. The description field 34 holds a stream descriptor for the encrypted data. The stream descriptor also includes an encryption key value 50.

The encryption key value 50 may be used to decrypt the data stored in the stream. The stream descriptor for the encrypted data that is held in the description field 34 may be a tiny stream, a small stream, or a large stream.

Another example of the use of the type field 32 to specify special descriptions of data is shown in Figure 8. Figure 8 shows the stream descriptor 28 for a small scale transaction. A small scale transaction refers to an instance wherein changes to data in a database are recorded without directly changing the data until a sufficient number of other changes have occurred to warrant incurrence of the overhead associated with changing all of the affected data. The type field 32 specifies that the description field 34 holds data for a small scale transaction. The description field 34 holds a first stream descriptor 52 and a second stream descriptor 54. The first stream descriptor 52 describes an original stream of data. The second stream descriptor 54 describes a stream that specifies the changes that have been made to the original stream. The original stream of data is updated by implementing the changes held in the second stream.

Figure 9 shows an example of the stream descriptor 28 for a stream that holds replicated data. Data often must be replicated so that loss of the data will not be catastrophic. Specifically, there are selected data structures for which the system maintains multiple copies on disk. In such instances, the structures are copied to two different locations on the disk. The stream descriptor 28 includes a first stream descriptor 56 and a second stream descriptor 58 in its description field 34. The first stream descriptor 56 describes a first stream holding a first copy of the data, and the second stream descriptor 58 describes a second stream holding another copy of the data. The type field 32 holds a value specifying that the stream includes replicated data.

The preferred embodiment of the present invention stores information about streams in a field descriptor 60 like that shown in Figure 10. The field descriptor 60 includes a stream ID field 62 that holds a stream ID. The stream ID is a four-byte long identification number that uniquely identifies the stream within an onode (onodes will be described in more detail below). The field descriptor 60 further includes a flags field 64 that holds flag bits. The final field of the field descriptor 60 is the stream descriptor 28 for the stream.

Streams are grouped according to related functionality into "onodes". An onode corresponds with the logical notion of an object and typically holds all of the streams that constitute a file, a directory or a subdirectory. Each onode includes information necessary to describe the variable-sized collection of streams that it includes.

Figure 11 is a diagram that illustrates the format of an onode 66. Each onode 66 holds streams of related functionality. Each onode 66 includes the following fields: a length field 68, a work ID field 70, a flags field 72, a class ID field 73 and a field 76 that holds an array of field descriptors 60 (such as depicted in Figure 10). The length field 68 holds a value that specifies the length of the onode, whereas the work ID field 70 holds an index into a work ID mapping array 104 (Figure 14), as will be described in more detail below. The work ID is four bytes in length. The flags field 72 (Figure 11) holds flag bits, and the class ID field 73 holds a class ID for the onode. Field 76 holds a packed array of field descriptors 60, that includes a field descriptor 60 for each of the streams held with the onode 66. The number of streams included in the array of field descriptors of field 76 may vary. Moreover, the length of each stream in the array of field descriptors of field 76 may vary. Accordingly, the onode 66 is a variable-sized structure. The variable-sized nature of onodes 66 helps to minimize internal fragmentation in allocation units on disk in disk storage 16.

Certain data about onodes 66 is not incorporated directly into the onode. Instead, this data is stored in separate streams as shown in Figure 12. Stream 78 holds a number of different types of status information regarding its associated onode 66 (Figure 11). The status information includes a time stamp held in field 86 that specifies the time that the onode 66 was created. Field 88 holds a time stamp that specifies the last time that the onode 66 was modified. Similarly, field 90 holds a time stamp that specifies the last time that the onode 66 was accessed. Field 92 holds a value that specifies the size of the onode 66, and field 94 holds a security descriptor for the owner of the onode. All of the information held in stream 78 is useful in administration of file data held in the associated onode 66.

A second stream 80 of status information is also shown in Figure 12. This stream 80 includes three fields 96, 98 and 100. Each onode 66 is visible in the global name space of the data processing system 10, and the global name space is a logical tree structure in which each onode other than the root onode has a parent node. Field 96 holds the work ID of the parent onode. Field 98 holds a universally unique ID (UUID) for the onode 66. Lastly, field 100 holds a class ID that specifies the class of the onode 66. Each onode 66 has a unique class associated with it. In particular, each onode 66 is an instance of an object of a particular class.

Two additional streams 82 and 84 of status information are also stored. Stream 82 holds the

name of the onode 66 relative to its parent, and stream 84 holds access control lists (used in security) for the onode.

Streams 78, 80, 82 and 84 are stored separately for at least two reasons. First, storing this information separately reduces the average size of the onodes 66. Streams 78, 80, 82 and 84 are not stored for each onode. As a result, the average size of the onodes decreases. Second, storing this information separately allows programmatic access to related groups of information without requiring complex code to retrieve the code from each onode 66.

Onodes 66 (Figure 11) are stored in onode bucket arrays 102 (Figure 13). The onode bucket array 102 is a variable-sized data structure that is composed of an array of fixed-sized buckets. The size of the buckets (e.g., 4K) depends on the architecture of the data processing system 10 but may match the page size of the system 10. The buckets are numbered 1 through N in the example of Figure 13. Each bucket in the array 102 contains a packed set of onodes 66. In the example of Figure 13, it can be seen that bucket 2 includes only a single onode, whereas bucket 1 includes two onodes and bucket 3 includes three onodes.

The onode bucket array 102 is used to minimize the amount of data that must be shuffled when file data is moved, deleted or inserted. The granularity of allocation and deallocation of blocks of disk space is fixed. In other words, memory blocks are allocated and deallocated in fixed-sizedd buckets. If, instead, a variable-sized structure was employed for storing file data, the granularity of allocation would not be fixed, and the granularity of allocation could become excessively large.

For efficiency, the operating system 24 (Figure 1) stores related onodes 66 (Figure 11) in the same buckets in the array 102 (Figure 14) or in buckets that are in close proximity to each other on disk in disk storage 16. This storage strategy minimizes seek time in finding onodes 66 on disk. In general, files that are typically accessed together are placed into the same bucket. Examples of files that are accessed together include files in a common subdirectory or directory.

The onode bucket array 102 (Figure 13) is accessed as a stream. The system 10 typically stores multiple onode bucket arrays 102 and, thus, stores multiple streams holding the onode bucket arrays. Internally, all references between onodes are based on work IDs. Locating an onode 66 (Figure 11) within the bucket of an array 102 requires looking for a work ID for the onode. The work ID mapping array 104 (Figure 14) maps from a work ID to a bucket number of the onode bucket array 102. The numbered bucket is then searched for an onode with a matching work ID. In particular, the work ID of a particular onode 66 serves an index into the work ID mapping array 104. The entry at the specified index identifies the bucket number that holds the onode 66. For example, as shown in Figure 14, entries 106 and 108 of the work ID mapping array 104 hold values specifying that the onodes 66 having the associated work IDs are held in bucket 1.

Each onode 66 (Figure 11) includes a name index stream that is stored among the streams held in the array of field descriptor 76. The name index stream holds a B-tree index for the stream descriptors contained within the onode 66. The name index stream uses the stream ID of a stream as a key for locating the stream descriptor held within the onode 66. The B-tree index is described in more detail in co-pending application entitled "Efficient Storage of Objects in a File System", Serial No. _____, which was filed on even date herewith and is assigned to a common assignee. The disclosure of that co-pending application is explicitly incorporated by reference herein.

Just as related collection of streams are collected into onodes 66 (Figure 11), related collections of onodes are collected into data structures known as object store catalogs. The work ID of an onode 66 serves as a basis for identifying any onode within an object store catalog. An object store catalog is described by an object store catalog onode 110, such as shown in Figure 15. The object store catalog onode 110 includes the length field 68, the work ID field 70, flags field 72 and an array of field descriptors 76 as are found in other onodes 66. The array of field descriptors 76 includes field descriptors in which are stored stream descriptors 112, 114 and 116.

Stream descriptor 112 describes the work ID mapping array stream 104. Stream descriptor 114 describes the onode bucket array stream 102, and stream descriptor 116 describes the name index stream 117. Since the work ID mapping array stream 104 and the onode bucket array stream 102 store related data, they are stored in a common onode (i.e., the object store catalog onode 110). As such, the object store catalog onode 110 provides a vehicle for relating the work ID mapping array stream 104, the onode bucket array stream 102 and the name index stream 117. The object store catalog onode 110 is stored within the group of onodes that constitute the catalog at work ID 0. Work ID 0 is, by definition, always located in the first element of the onode bucket array stream 102. The first bucket at work ID 0 is a distinguished bucket having a known location. The data structures stored within the bucket are, thus, easily and simply accessed.

While the present invention has been described with reference to a preferred embodiment

thereof, those skilled in the art will appreciate that various changes in detail and form may be made without departing from the present invention as defined in the appended claims.

Claims

1. In a data processing system having disk storage and a processing means for running an operating system, a method comprising the steps of:
   (a) storing data in the disk storage in a first variable-sized stream data structure for holding logically contiguous data types;
   (b) storing meta-data in the disk storage in a second variable-sized stream data structure for holding logically contiguous data bytes; and
   (c) for each of the stream data structures, storing a stream descriptor that includes a type identifier that identifies how the stream data structure is stored on disk in disk storage.

2. The method recited in claim 1 wherein the step of storing meta-data in the disk storage in the second variable-sized stream data structure further comprises the step of storing an index to stream data structures that hold related data in the disk storage in the second variable-sized stream data structure.

3. The method recited in claim 2 wherein the step of storing the index to stream data structures that hold related data in the disk storage in the second variable-sized data structure further comprises the step of storing a B-tree index to stream data structures that hold related data in the disk storage in the second variable-sized stream data structure.

4. The method recited in claim 1 wherein the step of storing a stream descriptor for each of the stream data structures further comprises the step of storing a stream descriptor for each of the stream data structures in the respective first and second variable-sized stream data structures.

5. In a data processing system having disk storage, a method comprising the steps of:
   (a) storing streams of data in the disk storage in stream data structures comprising logically contiguous bytes of data;
   (b) storing a stream descriptor for each stream data structure in the disk storage with each stream descriptor including a type identifier that describes how the stream data structure is stored in the disk storage; and
   (c) storing stream data structures which store related data and their associated stream descriptors in a first variable-sized data structure in the disk storage.

6. The method recited in claim 5 wherein the step of storing streams of data in the disk storage in stream data structures further comprises the steps of storing at least one stream of meta-data in the disk storage in a stream data structure and storing at least one stream of data that is not meta-data in the disk storage in a stream data structure.

7. The method recited in claim 5 wherein the step of storing the stream descriptor for each stream data structure in the disk storage further comprises the step of storing the stream descriptor for each stream data structure in the disk storage in another stream data structure.

8. The method recited in claim 5 wherein the step of storing stream data structures which store related data and their associated stream descriptors in the variable-sized data structure in the disk storage further comprises the step of storing stream data structures which store data of a file and their associated stream descriptors in the first variable-sized data structure in the disk storage.

9. The method recited in claim 5 wherein the step of storing stream data structures which store related data and their associated stream descriptors in the variable-sized data structure in the disk storage further comprises the step of storing stream data structures which store data of a directory and their associated stream descriptors in the first variable-sized data structure in the disk storage.

10. The method recited in claim 5 wherein the step of storing stream data structures which store related data and their associated stream descriptors in the variable-sized data structure in the disk storage further comprises the step of storing stream data structures which store data of a subdirectory and their associated stream descriptor in the first variable-sized data structure in the disk storage.

11. The method recited in claim 5, further comprising the step of storing stream data structures that are not stored in the first variable-sized data structure and that store related data and their associated stream descriptors in a second variable-sized data structure in the disk stor-

age.

12. The method recited in claim 11, further comprising the step of storing the first variable-sized data structure and the second variable-sized data structure in an array of fixed-sized buckets of disk space in the disk storage.

13. The method recited in claim 12, further comprising the step of storing the array of fixed-sized buckets of disk space in the disk storage in a stream data structure.

14. In a data processing system having disk storage, a method comprising the steps of:
(a) storing streams of data in the disk storage in stream data structures and said streams comprising logically contiguous bytes of data;
(b) storing a stream descriptor for each stream data structure in the disk storage, each stream descriptor including a type identifier that describes how the stream data structure is stored in the disk storage;
(c) storing stream data structures which store related data and their associated stream descriptors in a variable-sized data structure in the disk storage; and
(d) storing groups of the variable-sized data structures which store related data in a catalog data structure.

15. The method recited in claim 14 wherein the step of storing streams of data in the disk storage in stream data structures further comprises the steps of storing at least one stream of meta-data in the disk storage in a stream data structure and storing at least one stream of data that is not meta-data in the disk storage in a stream data structure.

16. The method recited in claim 14 wherein the step of storing the stream descriptor for each stream data structure in the disk storage further comprises the step of storing the stream descriptor for each stream data structure in the disk storage in another stream data structure.

17. The method recited in claim 14 wherein the step of storing stream data structures which store related data and their associated stream descriptors in the variable-sized data structure in the disk storage further comprises the step of storing stream data structures which store data of a file and their associated stream descriptors in the first variable-sized data structure in the disk storage.

18. The method recited in claim 14 wherein the step of storing stream data structures which store related data and their associated stream descriptors in the variable-sized data structure in the disk storage further comprises the step of storing stream data structures which store data of a directory and their associated stream descriptors in the first variable-sized data structure in the disk storage.

19. The method recited in claim 14 wherein the step of storing stream data structures which store related data and their associated stream descriptors in the variable-sized data structure in the disk storage further comprises the step of storing stream data structures which store data of a subdirectory and their associated stream descriptor in the first variable-sized data structure in the disk storage.

20. The method recited in claim 14, further comprising the step of storing stream data structures that are not stored in the first variable-sized data structure which store related data and their associated stream descriptors in a second variable-sized data structure in the disk storage.

21. The method recited in claim 20, further comprising the step of storing the first variable-sized data structure and the second variable-sized data structure in an array of fixed-sized buckets of disk space in the disk storage.

22. The method recited in claim 21, further comprising the step of storing the array of fixed-sized buckets of disk space in the disk storage in a stream data structure.

23. The method recited in claim 22, further comprising the step of storing a stream descriptor for the stream data structure that stores the array of fixed-sized buckets of disk space in one of the variable-sized data structures which storage related data.

24. The method recited in claim 22, further comprising the step of storing the catalog data structure in a predetermined fixed-sized bucket of the array of fixed-sized buckets of disk space.

25. In a data processing system having disk storage, a method comprising the steps of:
(a) storing streams of data in the disk storage in stream data structures, said streams comprising logically contiguous bytes of data;

(b) storing a stream descriptor for each stream data structure in the disk storage;
(c) storing stream data structures which store related data and their associated stream descriptors in respective variable-sized data structures in the disk storage, each variable-sized data structure having an associated identifier;
(d) storing the variable-sized data structures in an array of fixed-sized buckets of disk space, having associated identifiers, in the disk storage;
(e) storing a mapping structure holding entries that specify a bucket identifier for a bucket in the array in the disk storage, said entries being indexed by the identifiers of the variable-sized data structures; and
(f) using the mapping structure to locate one of the variable-sized data structures in the array given its identifier.

26. The method recited in claim 25 wherein the array is stored in the disk storage in one of the stream data structures and a stream descriptor for the stream storing the array is stored in disk storage.

27. The method recited in claim 26 wherein the mapping structure is stored in the disk storage in one of the stream data structures and a stream descriptor for the stream storing the mapping structure is stored in disk storage.

28. The method recited in claim 27 wherein the stream descriptor for the stream storing the array and the stream descriptor for the stream storing the mapping structure are stored in a selected one of the variable-sized data structures.

29. The method recited in claim 27 wherein the selected variable-sized data structure is stored in one of the buckets of the array.

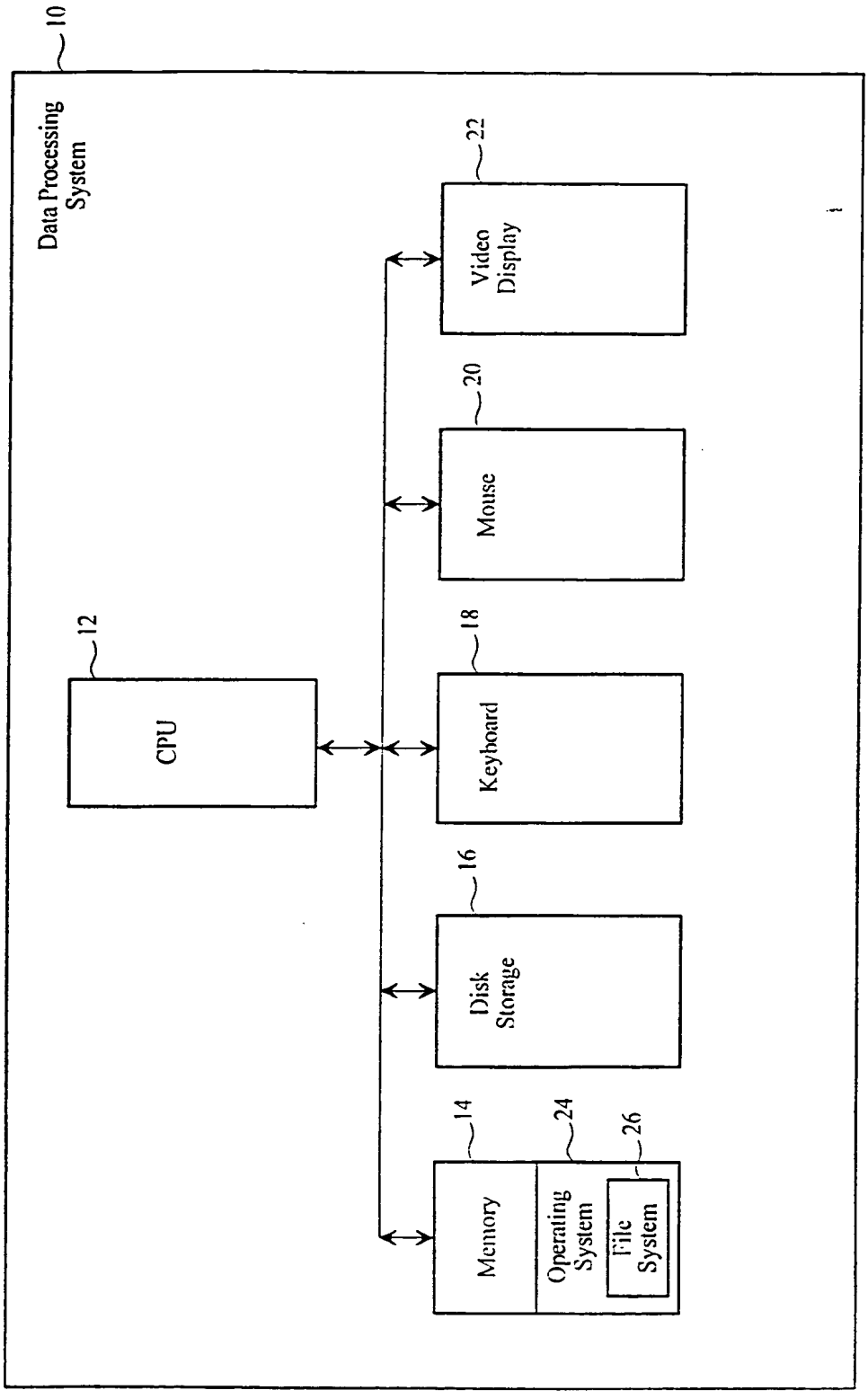
10
Data Processing System
22
Video Display
20
Mouse
12
CPU
18
Keyboard
16
Disk Storage
14
Memory
24
Operating System
26
File System

Figure 1

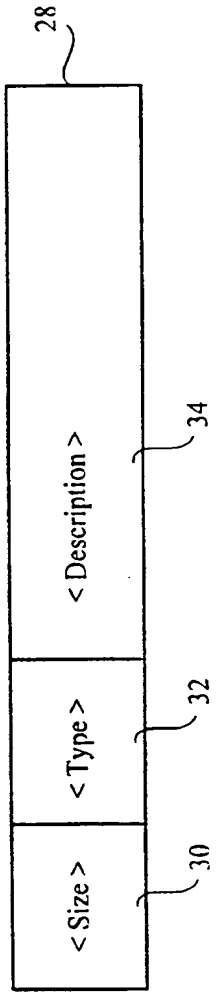

Figure 2

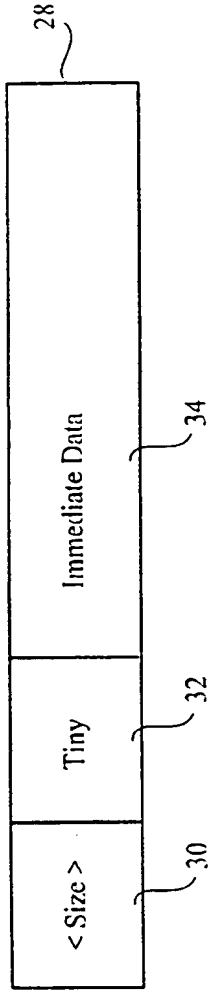

Figure 3

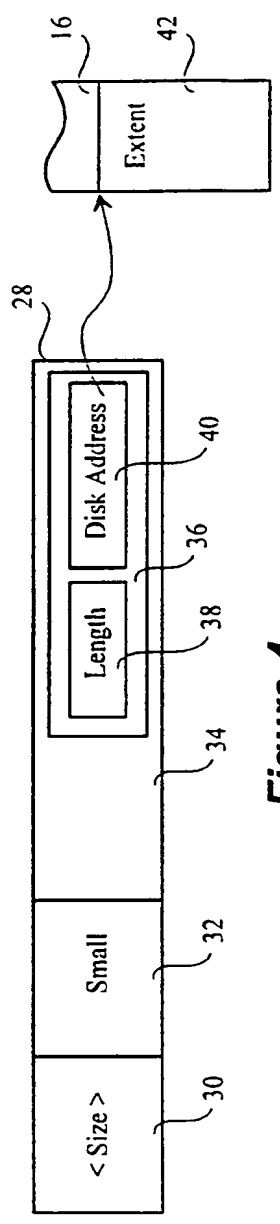

Figure 4

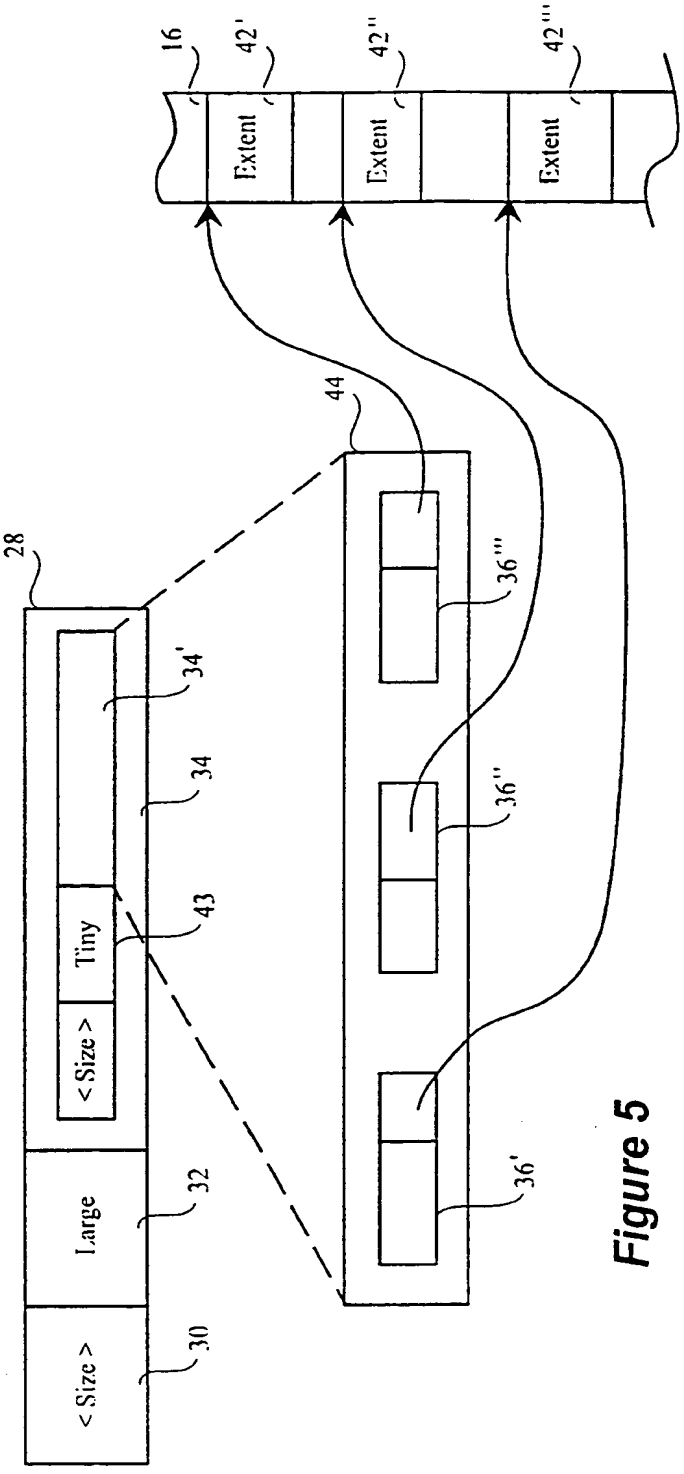

Figure 5

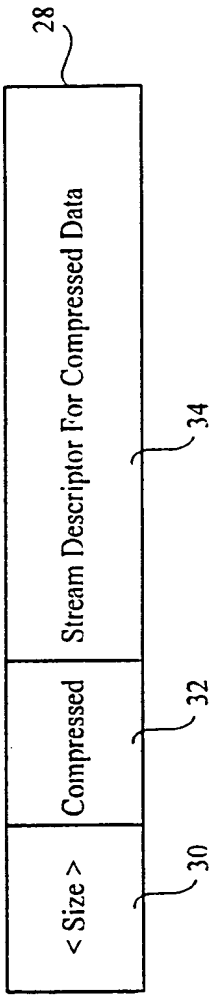

Figure 6

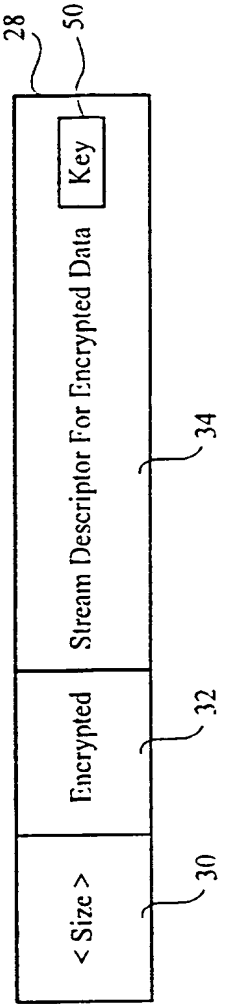

Figure 7

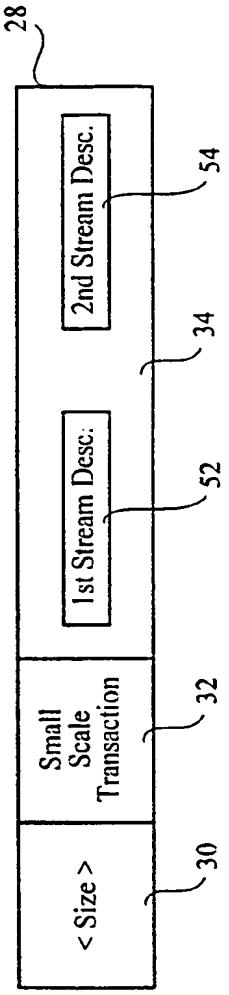

Figure 8

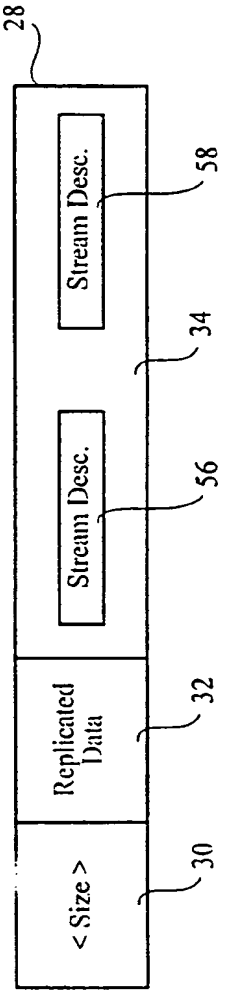

Figure 9

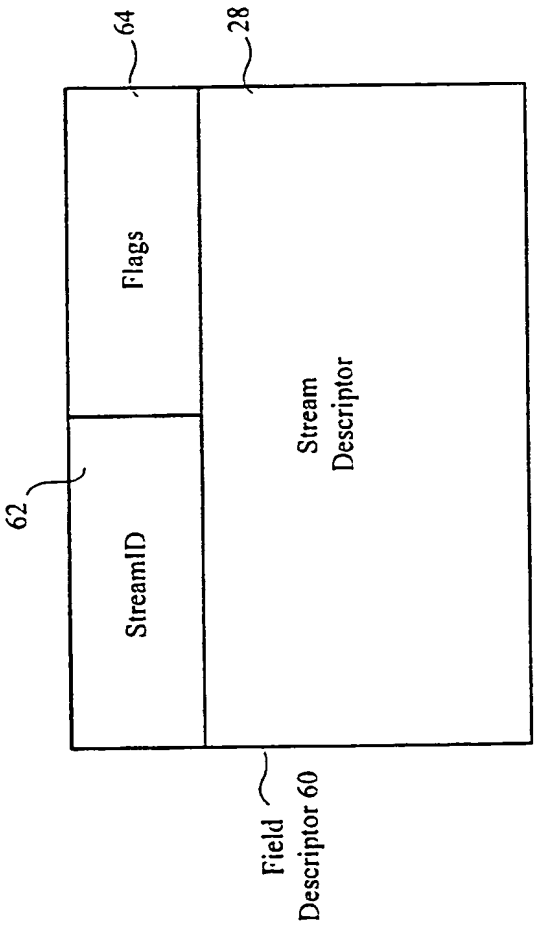
64
28
Flags
Stream
Descriptor
62
StreamID
Field
Descriptor 60

Figure 10

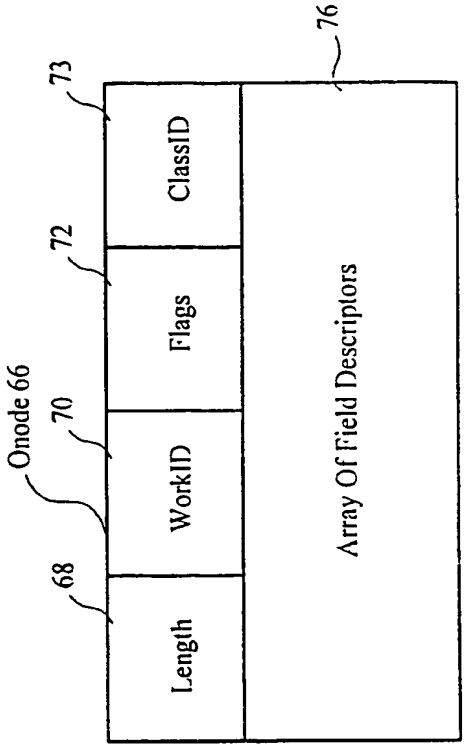
Onode 66
68
Length
70
WorkID
72
Flags
73
ClassID
76
Array Of Field Descriptors

Figure 11

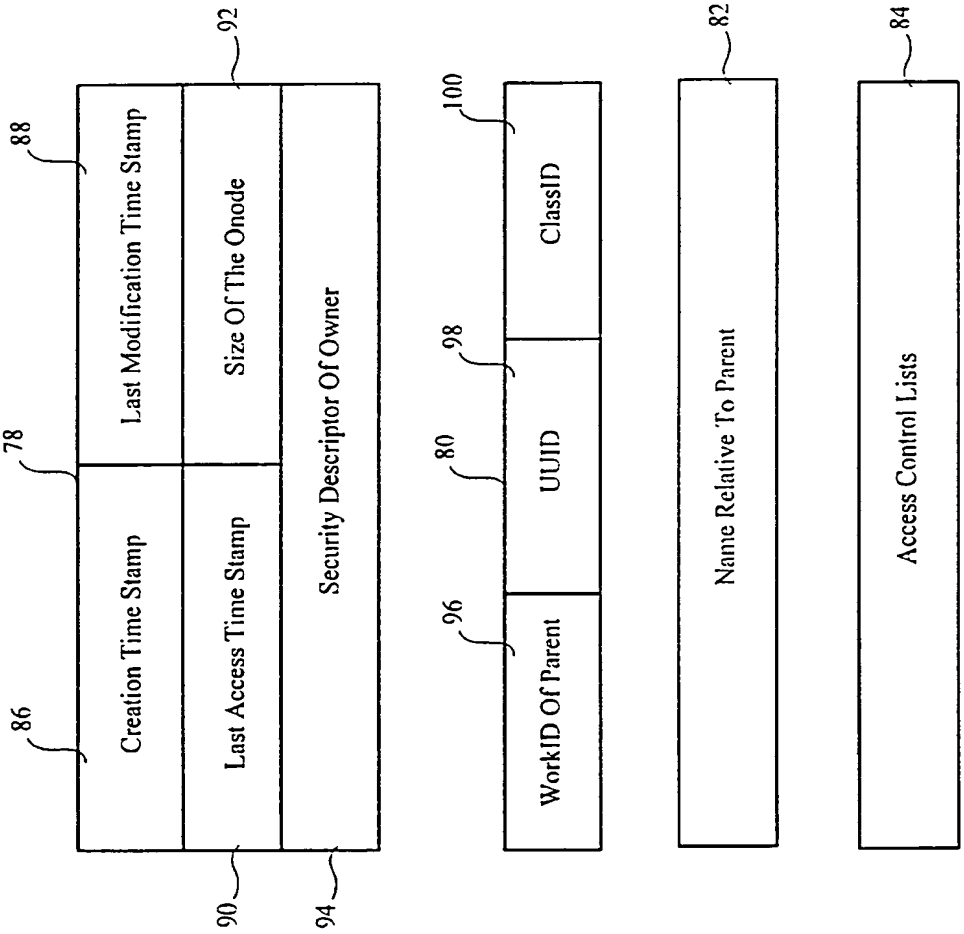
86
78
88
92
Creation Time Stamp
Last Access Time Stamp
Last Modification Time Stamp
Size Of The Onode
Security Descriptor Of Owner
90
94
96
80
98
100
WorkID Of Parent
UUID
ClassID
82
Name Relative To Parent
84
Access Control Lists

Figure 12

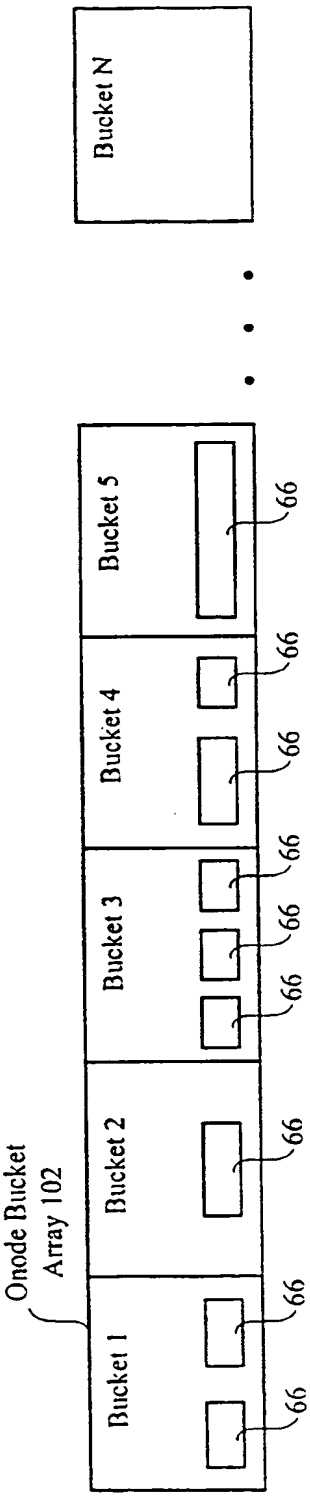
Onode Bucket Array 102
Bucket 1
Bucket 2
Bucket 3
Bucket 4
Bucket 5
Bucket N
66

Figure 13

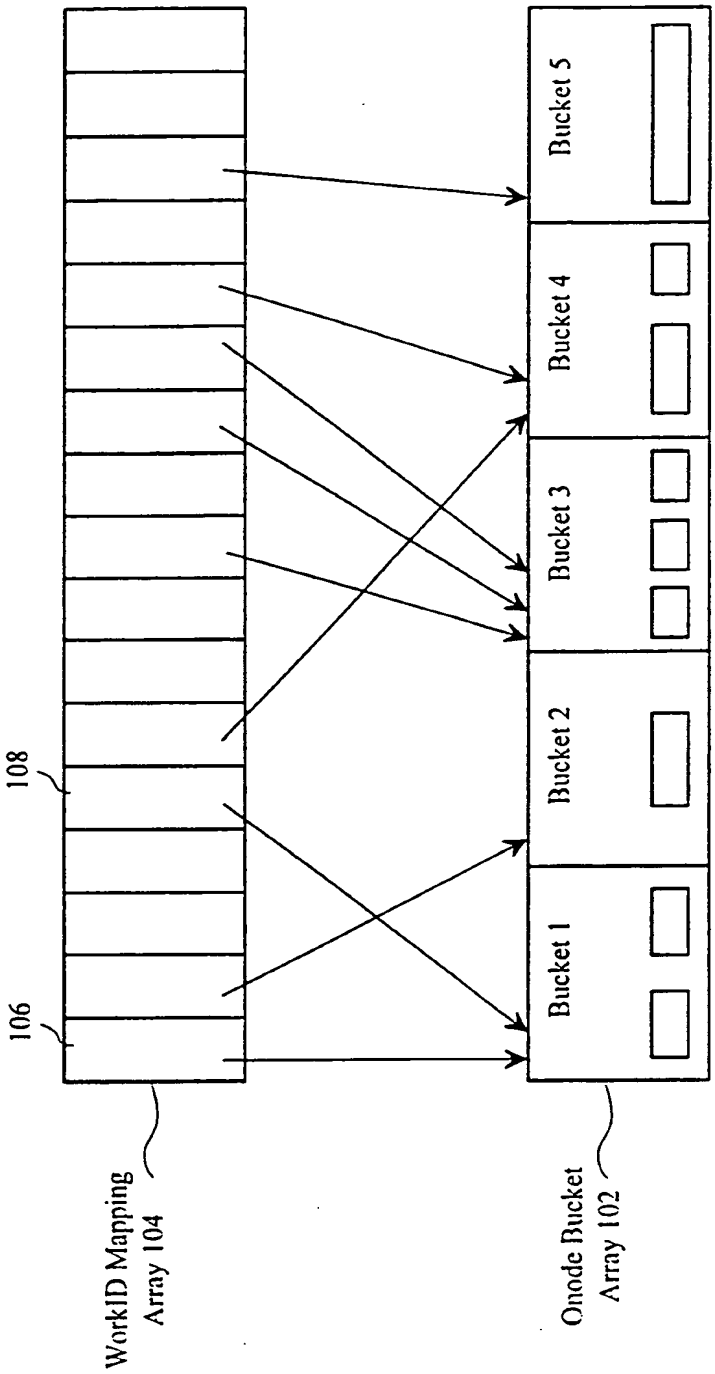
Bucket 5
Bucket 4
Bucket 3
Bucket 2
Bucket 1
108
106
WorkID Mapping
Array 104
Onode Bucket
Array 102

Figure 14

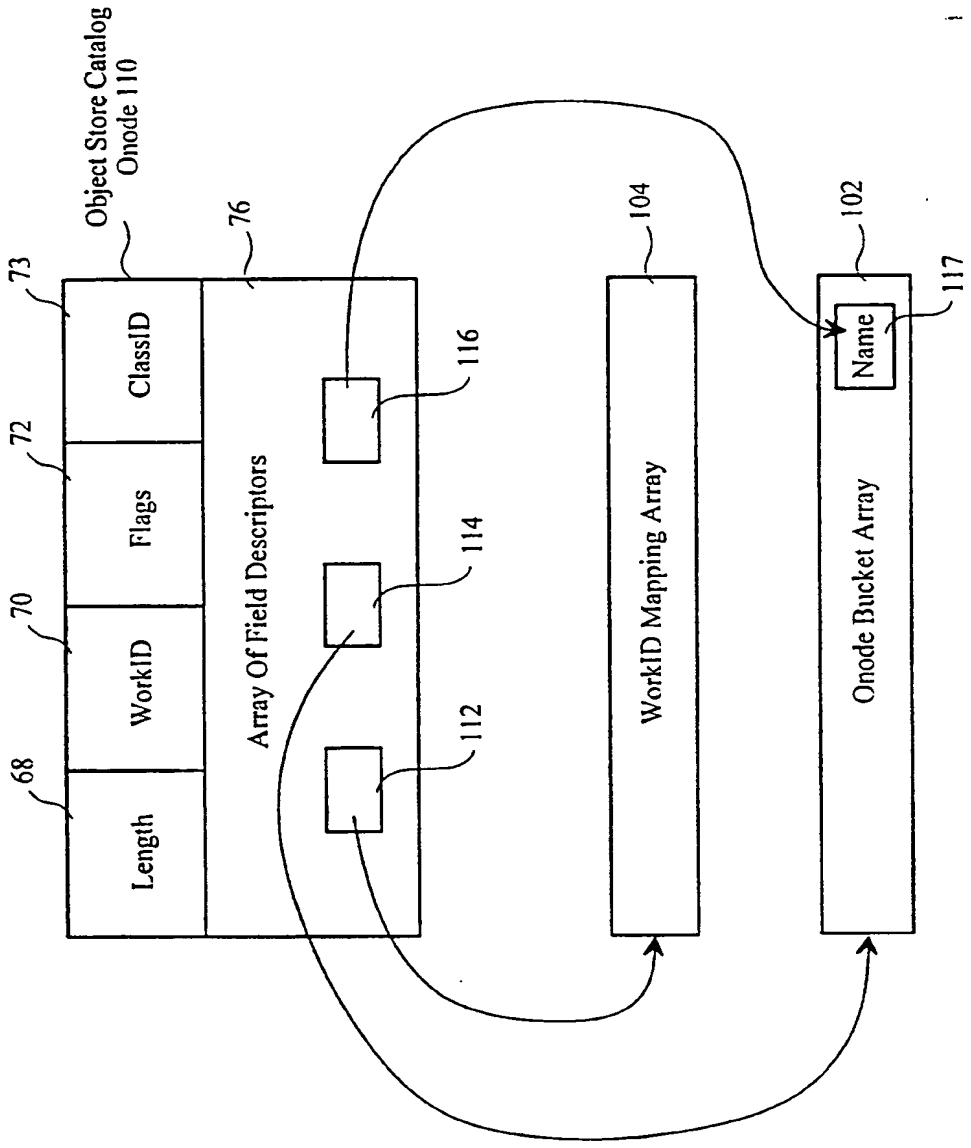
Object Store Catalog
Onode 110
73
ClassID
72
Flags
70
WorkID
68
Length
76
Array Of Field Descriptors
116
114
112
104
WorkID Mapping Array
102
Name
117
Onode Bucket Array

Figure 15

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 94 11 0018

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
X	EP-A-0 347 032 (IBM)	1,4,5, 7-11,14, 16-18,20	G06F3/06
A	* figures 1-3 * * page 3, line 19 - line 40 * * page 4, line 16 - page 5, line 16 *	25,26	
X	IBM TECHNICAL DISCLOSURE BULLETIN., vol.34, no.5, October 1991, NEW YORK US pages 422 - 423 'TOOL ,OT EXAMINE THE EFFICIENCY OF FILE PLACEMENT ON DISK'	1,2,14, 15	
A	* the whole document *	5,6	
A	IBM TECHNICAL DISCLOSURE BULLETIN., vol.29, no.12, May 1987, NEW YORK US pages 5256 - 5260 'ORDER-PRESERVING HASHING USING POSITIONAL CHARACTER TRANSITION FREQUENCIES'	14, 20-27,29	

TECHNICAL FIELDS SEARCHED (Int.Cl.6)

G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	21 October 1994	Weiss, P

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.